On Evaluating the Generalization of LSTM Models in Formal Languages

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber

{msuzgun@college, belinkov@seas, shieber@seas}.harvard.edu

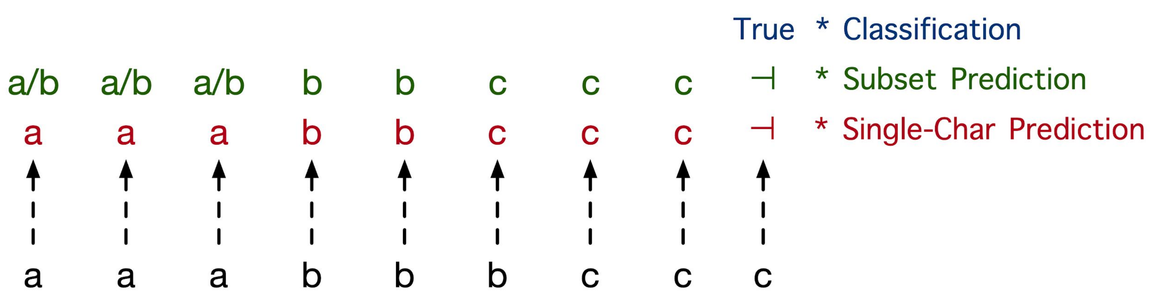
HARVARD
John A. Paulson
School of Engineering
and Applied Sciences

Introduction

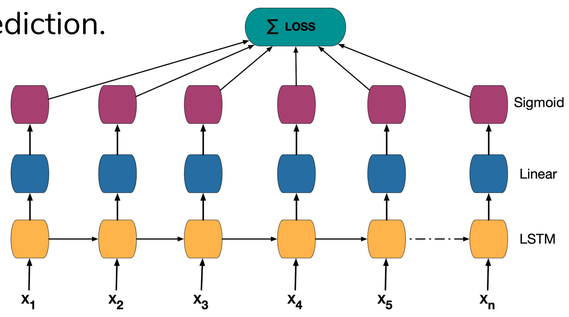
- Long Short-Term Memory (LSTM) networks, a popular variant of Recurrent Neural Networks (RNNs), are often used in NLP tasks.
- The classes of languages that can be learned, empirically or theoretically, by LSTM models are still unknown.
- But the notions of "*learning*" and "*generalization*", from a neural network's perspective, are ambiguous.
- The contributions of this work are fourfold:
  - (1) Considering three simple formal languages, namely a^nb^n, $a^nb^nc^n$, and $a^nb^nc^nd^n$, we explore three parameters that influence the inductive learning capabilities of LSTMs in formal languages:
    - (i) *Training length distribution*,
    - (ii) *Training length window*, and
    - (iii) *Network capacity*.
  - (2) We propose a new fine-grained evaluation scheme.
  - (3) We show that LSTM networks trained with the same training parameters but with different weight initializations can have different generalizations even though they all converge to similar loss values.
  - (4) We demonstrate that LSTM models learn to develop counting mechanisms to recognize the aforementioned languages.

The Sequence Prediction Task

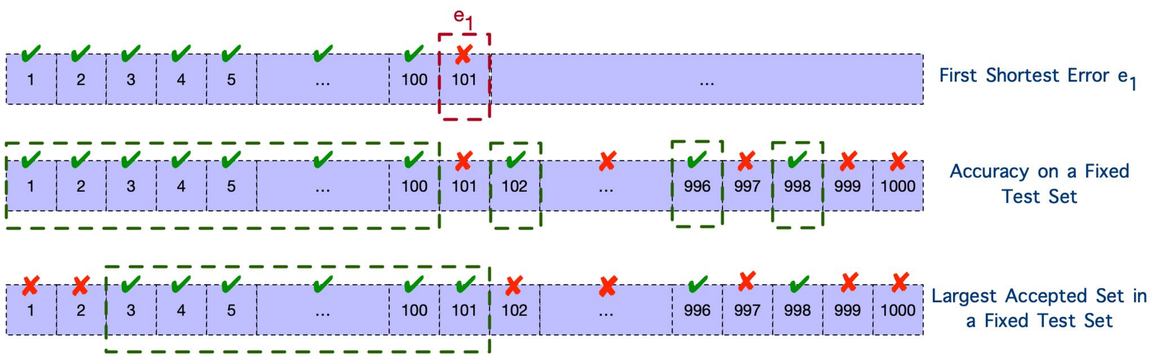
- Three standard methods of defining a language learning platform:



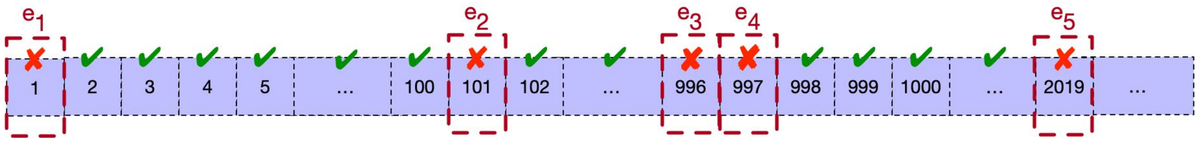
- We adapt the second method: Subset Prediction.
- Languages:
  - CFL: $a^n b^n \Rightarrow (a/b)^n b^{n-1}\dashv$
  - CSL: $a^nb^nc^n \Rightarrow (a/b)^n b^{n-1}c^n\dashv$
  - CSL: $a^nb^nc^nd^n \Rightarrow (a/b)^n b^{n-1}c^nd^n\dashv$



- There are, however, various methods of evaluation of the performance of a neural network in a formal language learning task:

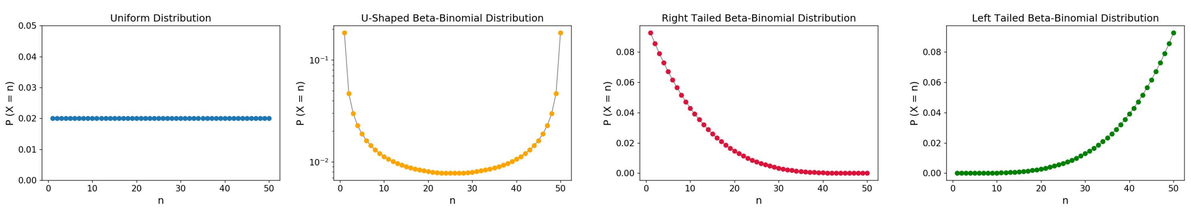


- Our evaluation method (❁):



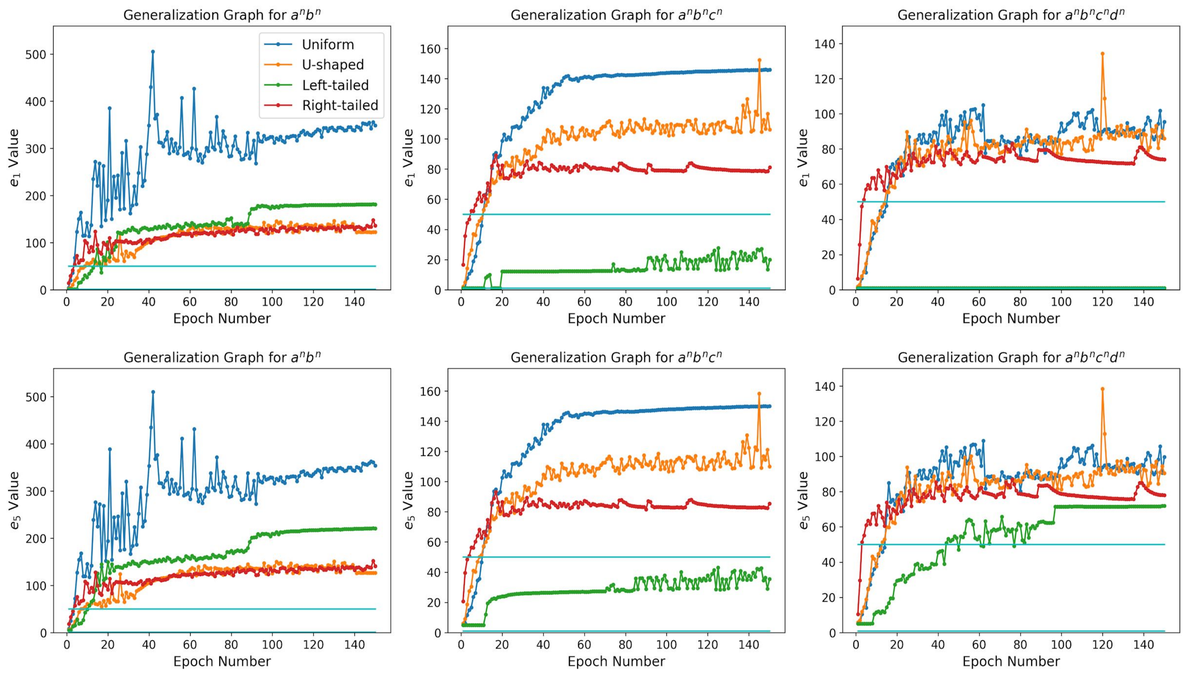
Experimental Setup

- Training and testing alternated: At each epoch, we present 1,000 samples to our model, then freeze all the weights, exhaustively enumerate all the sequences in the language, and finally determine the first *k*-shortest sequences whose outputs are incorrectly predicted.
- **Length Distributions:** Beta-Bin (Left-tailed, Right-tailed, U-shaped) and Uniform.

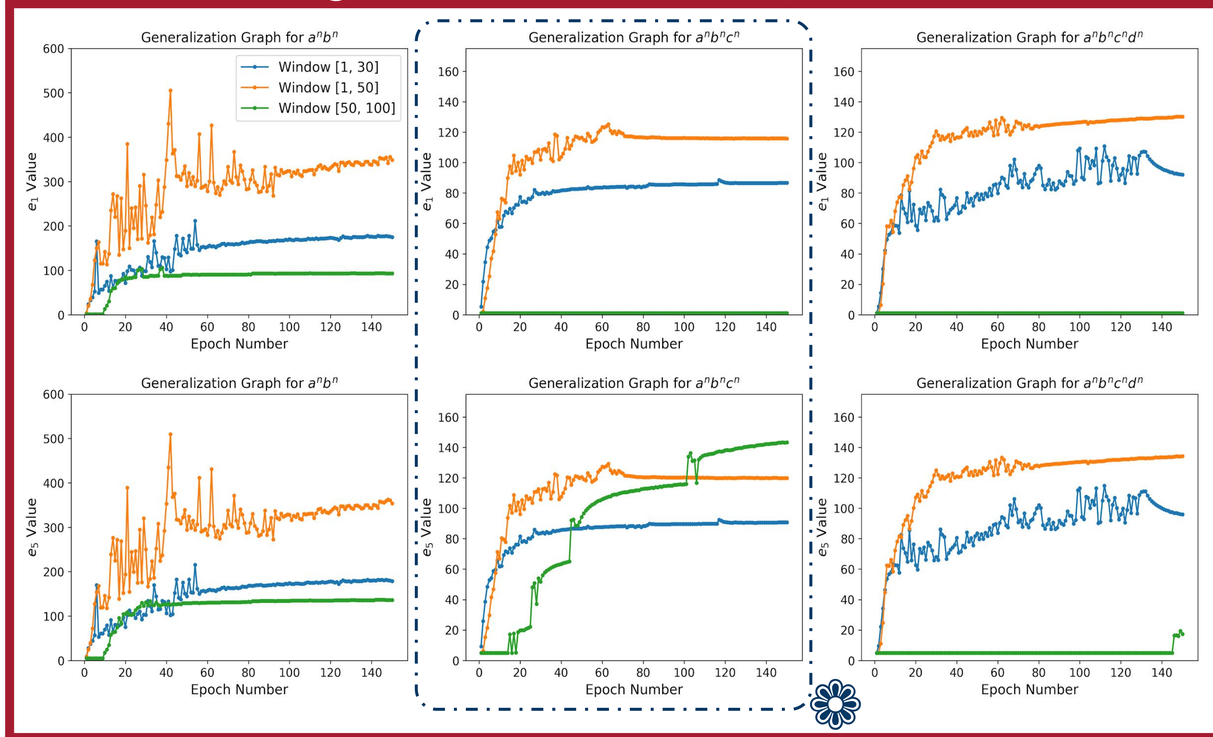


- **Length Windows:** [1, 30], [1, 50], and [50, 100].
- **Hidden Units**: 1,2,3,36 for a^nb^n; 2,3,4,36 for $a^nb^nc^n$; and 3,4,5,36 for $a^nb^nc^nd^n$.

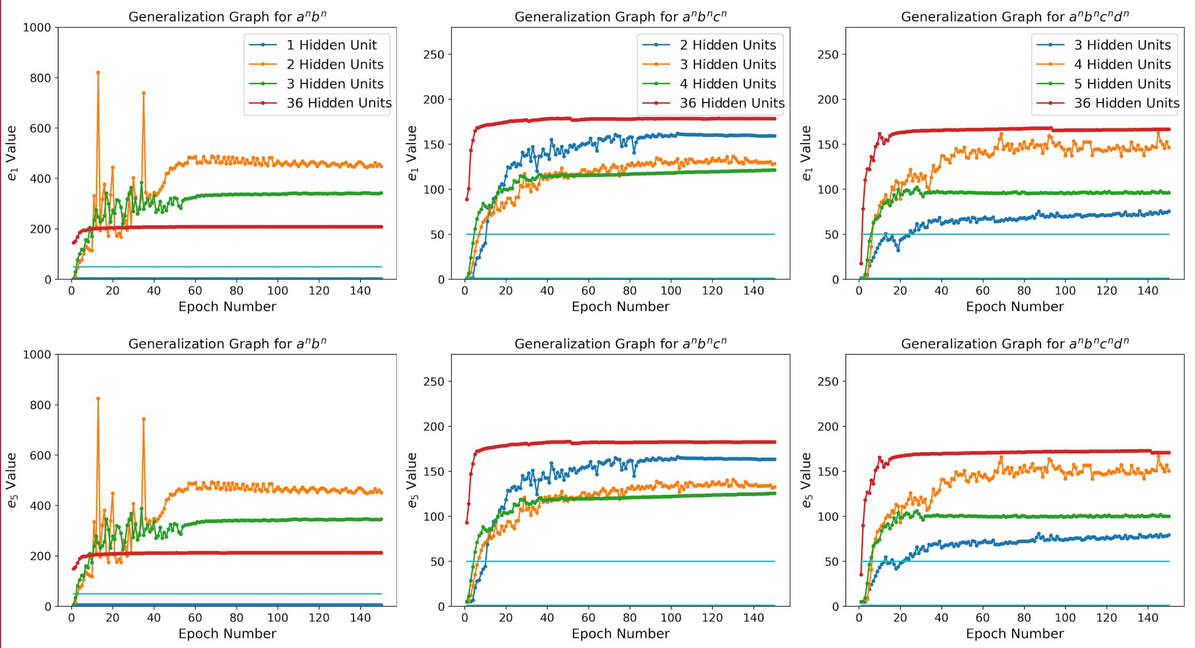
Length Distributions: Uniform and U-Shaped Beta-Bin Win!



Length Window: The Wider, the Better!

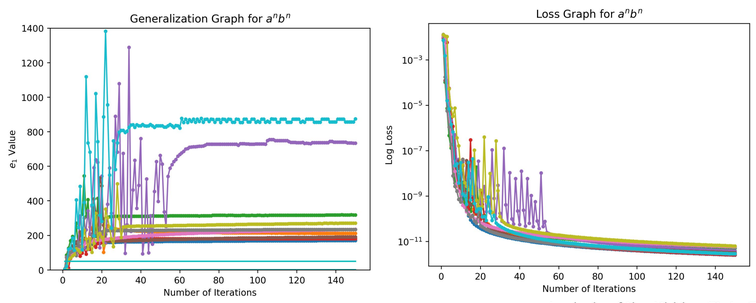


Network Capacity: More Hidden Units, Better Stability!

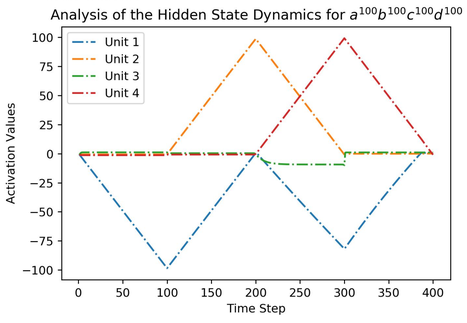


Discussion & Analysis: Weight Initialization and Counting

- There is a disconnection between loss convergence and generalization capabilities.



- LSTM networks organize their hidden state structure in such a way that certain hidden units learn how to count up and down upon the subsequent encounter of some characters.
- In fact, some hidden units appear to cooperate together to count.



If you want to learn more, please visit: https://github.com/suzgunmirac/lstm-eval